

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.

1300 I STREET, N. W.
WASHINGTON, DC 20005-3315

202 • 408 • 4000
FACSIMILE 202 • 408 • 4400

WRITER'S DIRECT DIAL NUMBER:

(202) 408-4470

April 6, 2000

TOKYO
011 • 813 • 3431 • 6943
BRUSSELS
011 • 322 • 646 • 0353

Assistant Commissioner of Patents
Washington, DC 20231

Re: New U.S. Patent Application
Title: METHOD AND SYSTEM FOR ESTABLISHING
TRUST IN DOWNLOADED PROXY CODE
Inventors: Peter C. Jones, Robert W. Scheifler and
James H. Waldo
Attorney Docket No. 06502.0254

Sir:

We enclose the following papers for filing in the United States Patent and Trademark Office in connection with the above-referenced patent application.

1. Application - 28 pages, including 7 independent claims and 14 claims total.
2. Drawings - 5 sheets of informal drawings (Figs. 1 - 4B).

This application is being filed under the provisions of 37 C.F.R. §§ 1.53(b) and 1.53(f). Applicants await notification from the Patent and Trademark Office of the time set for filing the Declaration. Please accord this application a serial number and filing date.

04/06/00
JC535 U.S. PTO
ATLANTA
653 • 6400
PALO ALTO
650 • 849 • 6600

JC678 U.S. PTO
09/543908
04/06/00

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.

Assistant Commissioner of Patents

April 6, 2000

Page 2

No filing fee is being paid at this time. With the exception of the filing fee, the Commissioner is hereby authorized to charge any fees due, including fees under 37 C.F.R. § 1.16 or § 1.17, during the pendency of this application to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER L.L.P.

By: 

Michael L. Kiklis

Reg. No. 38,939

MLK:laf
Enclosures

United States Patent Application
of
Peter C. Jones
Robert W. Scheifler
and
James H. Waldo
for
Method and System for
Establishing Trust in Downloaded Proxy Code

LAW OFFICES

FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

RELATED APPLICATION

The present application is related to U.S. Provisional Patent Application No. 60/128,406, entitled "RMI Security," filed April 8, 1999, which is relied upon and is incorporated herein by reference.

FIELD OF THE INVENTION

The present invention relates generally to data processing systems and, more particularly, to using downloaded code to provide secure communication between a client and a remote service in a distributed system.

BACKGROUND OF THE INVENTION

Today's distributed systems can be made up of various components, including both hardware and software. Nodes in distributed systems typically communicate via a network such as the Internet. One means of communication between programs in a distributed system is downloading code from one program to another. For example, a client (e.g., a program running on a node in a distributed system) can access a service running on a remote node by downloading code from the remote service.

A "service" refers to a resource, data, or functionality that can be accessed by a user, program, device, or another service and that can be

computational, storage related, communication related, or related to providing access to another user. Examples of services include devices, such as printers, displays, and disks; software, such as applications or utilities; information, such as databases and files; and users of the system.

5 In a distributed system implemented using the Java™ programming language, services appear programmatically as objects, with interfaces that define the operations available from the service. In the Java™ programming language, a "class" provides a template for the creation of "objects" (which represent items or instances manipulated by the system) having characteristics of that class. Thus, a class defines the type of an object. Methods associated with a class are generally invoked on the objects of the same class or subclass. The Java™ programming language is described in The Java™ Language Specification by James Gosling, Bill Joy, and Guy Steele, Addison-Wesley, 1996, which is incorporated herein by reference.

10
15 In a distributed system that depends on downloaded code, there is the danger that the downloaded code may be untrustworthy. For example, the code could carry a virus or disguise its true source, causing the user of the downloaded code to disclose confidential information to an unknown party. Therefore, it is desirable to enable users in a distributed system to confirm that downloaded code is trustworthy.

SUMMARY OF THE INVENTION

5 A system consistent with the present invention enables a user in a distributed system to determine whether downloaded code is trustworthy before using the downloaded code to communicate with others in the distributed system. For example, if a client downloads code from a service, the client can verify that both the service and the downloaded code are trustworthy before using the code to communicate with the service. "Trustworthy" code is code that the client knows will enforce the client's security constraints (e.g., mutual authentication, confidentiality, and integrity) when communicating with the service.

10 In accordance with one implementation of the present invention, code is downloaded from a server, and a set of constraints to implement secure communication with the server is determined. Secure code is then used to verify that the downloaded code will enforce the set of constraints when the downloaded code is used to communicate with the server.

15 In accordance with another implementation of the present invention, a first proxy containing code for communication purposes is downloaded, and a second proxy containing code for communication purposes is obtained from the first proxy. A trustworthiness verification routine is used to determine whether the second proxy is trustworthy, and when it has been determined that the second proxy is trustworthy, the second proxy is used to determine whether a server is

trustworthy. When it has been determined that the server is trustworthy, a trustworthiness verification routine is requested from the server by using the second proxy and this verification routine is then used to identify the trustworthiness of the first proxy. When it has been determined that the second proxy is trustworthy, that the server is trustworthy, and that the first proxy is trustworthy, the first proxy is used to invoke a method on the server.

BRIEF DESCRIPTION OF THE DRAWINGS

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

Fig. 1 depicts a distributed system suitable for practicing methods and systems consistent with the present invention;

Fig. 2 depicts the logical interaction between a client and a service once the client has downloaded a proxy;

Fig. 3 is a high-level flow chart of an exemplary method for establishing trust in a downloaded proxy in accordance with methods and systems consistent with the present invention; and

Figs. 4A and 4B are flow charts depicting the steps of the verifyProxyTrust method for establishing trust in a downloaded proxy.

DETAILED DESCRIPTION OF THE INVENTION

5 A distributed system suitable for practicing methods and systems
consistent with the present invention can be implemented using the Jini™
architecture. A Jini™ system is built around one or more lookup services that list
the services available in the distributed system. When a service joins the
network, it uses a process called discovery to locate the lookup service or
services for the network. The service registers by passing a proxy object to each
lookup service. The proxy object is a Java™ object implementing the interfaces
of the corresponding service. When a client (e.g., a program) wishes to use a
10 service, the client interacts with the lookup service and downloads the proxy to
facilitate use of the service. The Jini™ architecture is described in more detail in
Arnold, The Jini™ Specification, Addison-Wesley (1999).

15 A system consistent with the present invention enables a program in a
distributed system to determine whether downloaded code is trustworthy before
using the downloaded code to communicate with other programs or services in
the distributed system. For example, if a client downloads code from a service,
the client can verify that both the service and the downloaded code are
trustworthy before using the code to communicate with the service.

"Trustworthy" code is code the client knows will enforce the client's security
constraints in communicating with the service.

These constraints can include, for example, integrity, anonymity, mutual authentication, delegation, and confidentiality. The "integrity" constraint ensures that messages will not be tampered with in transit, the "anonymity" constraint permits the identity of the client to stay unknown to the service, and "mutual authentication" refers to the client and service verifying their identities to one another. The "delegation" constraint allows the service to make calls to other computers using the client's identity, and the "confidentiality" constraint ensures that messages are private, e.g., by using encryption.

The constraints attached to a proxy are included in the serialized version of the proxy. As part of RMI system 122, an object, such as a proxy, is converted into a serialized version of itself before being passed. The serialized object contains enough information to enable the recipient to identify and verify the Java™ class from which the contents of the object were saved and to restore the contents to a new instance. Object serialization is explained in the Java™ Object Serialization Specification, available at <http://web2.java.sun.com/products/jdk/1.3/docs/guide/serialization/spec/serialTOC.doc.html>, which is incorporated herein by reference. When a client trusts the downloaded code to enforce the client's security constraints and trusts the service, the client can trust communicating through the proxy to the service and can trust the service to do what the client asks, and not something harmful.

Figure 1 depicts a distributed system 100 suitable for practicing methods and systems consistent with the present invention. Distributed system 100 includes a plurality of computers 102, 104, and 106, communicating via a network 108. Network 108 may be a local area network, wide area network, or the Internet. Computer 102 includes a memory 110, a secondary storage device 112, a central processing unit (CPU) 114, an input device 116, and a video display 118. Memory 110 includes a client program 142 and Java™ runtime system 120, which includes a Java™ remote method invocation (RMI) system 122.

Computers 104 and 106 may be similarly configured, with computer 104 including memory 124, secondary storage device 126, and CPU 128, and computer 106 including memory 130, secondary storage device 132, and CPU 134. Memory 124 of computer 104 includes a service 136, and memory 130 of computer 106 includes a lookup service 138 that defines the services available in one part of distributed system 100. Lookup service 138 contains one proxy object for each service within that part of distributed system 100. Each "proxy object" corresponds to a service in the distributed system and implements interfaces corresponding to methods available from a service.

For example, if service 136 were a database service, the corresponding proxy object 140 would implement database interfaces such as "find" or "sort."

To invoke methods on database service 136, client 142 asks lookup service 138

for a proxy to the database service. In response, lookup service 138 returns proxy object 140 corresponding to database service 136 to the client. Lookup services are explained in greater detail in U.S. Patent Application No.09/044,931, entitled "Dynamic Lookup Service in a Distributed System," which is incorporated herein by reference.

Figure 2 depicts the logical interaction between client program 142 and service 136 once the client has downloaded proxy 140. Proxy 140 implements one or more interfaces 202 corresponding to the methods 204 offered by service 136. Client 142 invokes methods on proxy 140. Proxy 140 processes method invocations, communicates with service 136 to execute the requests of client 142, and returns results. The protocol between proxy 140 and service 136 is not set by the Jini™ system; instead, it is determined by the service and its proxy. This enables a client to communicate with a service it may have never seen before without knowing or caring how the service works.

In this way, the Jini™ architecture relies on downloaded code to provide services in a dynamic and flexible way. Once a client downloads a proxy object, the client invokes methods on the proxy to communicate with the remote service. However, the proxy may be implemented using code that the client does not trust. Before sharing any critical data with the service, the client needs to establish trust in the proxy, the service, or both.

Figure 3 is a high-level flow chart of an exemplary method for establishing trust in a downloaded proxy in accordance with methods and systems consistent with the present invention. The exemplary method enables a client to use code that can be verified locally to communicate with a remote service. Using the locally-verified code, the client verifies the remote service and then asks the service whether the service trusts the downloaded proxy. If the client has verified the service and the service trusts the proxy, then the client has established transitive trust in the proxy.

With reference to figure 3, the exemplary method begins when the client downloads the proxy, P1, from a lookup service (step 302). The client then asks P1 for a second proxy to the service, P2, by invoking a method on P1 corresponding to a method on the service (step 304). The client examines the code in P2 to verify that P2 uses only trusted code (step 306). If the client verifies P2 (step 307), then the client uses P2 to authenticate the service by invoking a method on P2 (step 308). If the service is authenticated (step 309), the client then uses P2 to ask the service if the service trusts P1's code, by invoking a method on P2 to obtain a proxy verifier from the service, and then passing P1 to the proxy verifier (step 310). If the service trusts P1 (step 311), then the client can trust P1 (step 312). In this manner, the client uses local, trusted code to establish trust in a downloaded proxy.

The VerifyProxyTrust method

The exemplary method described above will now be explained using the Java™ RMI Security Subsystem, part of RMI system 122. The Security Subsystem defines classes and interfaces that ensure secure communication between remote objects. One class provided by the RMI Security Subsystem is the Security class, which includes the verifyProxyTrust method. The verifyProxyTrust method establishes trust in downloaded code and is defined as follows:

```
public static void verifyProxyTrust (  
    Object proxy,  
    SecurityConstraints constraints)  
    throws RemoteException;
```

The verifyProxyTrust method in RMI system 122 is called by client 142 once the client downloads proxy 140, before the client makes any other use of the proxy. As described below, the method will establish trust in the proxy by confirming that the proxy will correctly implement the RemoteSecurity interface, part of the Security class.

When a remote service instantiates its proxy, the service can include the RemoteSecurity interface, which provides methods enabling a client to attach security constraints to the proxy or to query the service to determine the service's security constraints. The RemoteSecurity interface is defined as follows:

```
public interface RemoteSecurity {  
    RemoteSecurity setClientConstraints (  
        SecurityConstraints constraints);  
}
```

```
SecurityConstraints getClientConstraints();
SecurityConstraints getServerConstraints (String name,
                                         Class[] parameterTypes);
    throws NoSuchMethodException, RemoteException;
boolean equalsIgnoreConstraints(Object obj);
}
```

The setClientConstraints method allows a client to make a copy of the proxy with a new set of constraints selected by the client. The getClientConstraints method returns the current client constraints attached to the proxy. The getServerConstraints method returns the server's constraints for a particular remote method implemented by the proxy.

As explained above, the constraints can include, for example, integrity, anonymity, mutual authentication, delegation, and confidentiality. When the verifyProxyTrust method determines that the proxy will enforce the client's constraints by correctly implementing the RemoteSecurity interface, the proxy is "trusted."

Figures 4A and 4B are flow charts showing the steps of the verifyProxyTrust method. When a client calls verifyProxyTrust, the call has as its parameters the proxy and the client's security constraints.

As shown in steps 402 to 410 of Figure 4A, the verifyProxyTrust method first determines whether the downloaded proxy is a secure RMI stub. A stub is one example of a proxy, created by the RMI System, that uses the RMI protocol to communicate with a service. A secure RMI stub is an instance of a class generated by the java.lang.reflect.Proxy class provided by the Java™

programming language. The methods of the Proxy Class include
Proxy.isProxyClass, which returns true if it is passed a proxy class that was
generated by the Proxy Class, Proxy.getInvocationHandler, which returns the
invocation handler associated with the proxy instance passed as its argument,
and Proxy.getProxyClass, which generates a java.lang.Class object for a proxy
given a class loader and an array of interfaces. These methods and the
java.lang.reflect.Proxy class are explained in further detail in the "Java™ 2
Platform, Standard Edition, v 1.3 API Specification," available at
<http://java.sun.com/products/jdk/1.3/docs/api/java/lang/reflect/Proxy.html>, and
incorporated herein by reference.

The verifyProxyTrust method calls the Proxy.isProxyClass method,
passing the class of the proxy as a parameter, to determine whether the proxy is
an instance of a trusted generated Proxy class (step 402). As described above,
the Proxy.isProxyClass method will return true if and only if the specified class
was dynamically generated to be a trusted proxy class.

If the Proxy.isProxyClass method returns true, the proxy is an instance of
a trusted class, and the verifyProxyTrust method tests the invocation handler of
the proxy to determine if it is an instance of a trusted class (step 404). Each
secure RMI stub has an invocation handler used to invoke methods on the proxy.
The verifyProxyTrust method calls the local Proxy.getInvocationHandler method,
passing the proxy as a parameter. The getInvocationHandler method returns the

invocation handler of the proxy, and the verifyProxyTrust method then calls the local instanceof operator to determine whether the proxy's invocation handler is an instance of a local trusted class,

SecureInvocationHandler. The local trusted class SecureInvocationHandler is specified in the Security class in RMI 122. The corresponding local calls made by verifyProxyTrust are:

```
Proxy.isProxyClass(proxy.getClass())  
handler = Proxy.getInvocationHandler(proxy)  
handler instanceof SecureInvocationHandler
```

As explained above, these methods are provided by the java.lang.reflect.Proxy class of the Java™ programming language.

If the proxy is an instance of a trusted class and the invocation handler is secure, then each socket factory instance contained in the invocation handler is checked (step 406). A socket is an end-point to a communication path between two processes in a network. A socket factory is an object that implements a method to create a new socket. Socket factories are described in more detail at <http://java.sun.com/products/jdk/1.3/docs/api/java/net/SocketImplFactory.html>. The class of each socket factory instance in the proxy is compared to a local list of trusted socket factory classes.

To obtain the list of trusted socket factory classes, the verifyProxyTrust method uses a local configuration database to obtain TrustVerifier.

TrustVerifiers are provided as part of the Java™ RMI Security Subsystem in RMI system 122, and are defined as follows:

```
public interface TrustVerifier {  
    boolean trustedConstraintClass(Class c);  
    boolean trustedPrincipalClass(Class c);  
    boolean trustedClientSocketFactoryClass(Class c);  
    boolean trustedProxy(Object proxy,  
                           SecurityConstraints constraints)  
        throws RemoteException;  
}
```

The local configuration database contains a list of TrustVerifier instances that implement methods for testing the security of a given proxy or class. The trustedClientSocketFactoryClass method will return true if the given class is a trusted socket factory class defined in RMI 122. For each socket factory instance in the proxy, the verifyProxyTrust method calls the trustedClientSocketFactoryClass method of each TrustVerifier instance, passing the socket factory class as a parameter. If at least one trustedClientSocketFactoryClass method returns true for each socket factory, the proxy is a secure RMI stub (step 408). If the proxy is not an instance of a trusted Proxy class, or if the invocation handler is not secure, or if the socket factories are not trusted, then the proxy is not a secure RMI stub (step 410).

Before continuing with the description of figure 4A, it is important to understand the difference between a unicast and an activatable stub. An RMI stub can be either unicast or activatable, depending on how it was exported by the service it represents. A unicast stub works until its service goes down. By

contrast, an activatable stub will still work after its service goes down because an activatable stub is capable of restarting the service, if necessary. Each activatable stub contains an Activation ID consisting of information needed for activating the stub's corresponding object, e.g., a remote reference to the object's activator and a unique identifier for the object. Activatable objects are explained in more detail in the Java™ 2 Platform, Standard Edition, v.1.2.2 API Specification, available at <http://www.java.sun.com/products/jdk/1.2/docs/api/java/rmi/activation/package-summary.html>, incorporated herein by reference.

If the proxy is activatable, i.e., the proxy contains an Activation ID, (step 412) then the verifyProxyTrust method takes additional steps to determine whether the proxy is a secure RMI activatable stub. The verifyProxyTrust method obtains an activator verifier by making a remote call, using the proxy's invocation handler, to the remote service's getActivatorVerifier method, passing the client constraints as parameters (step 414). The getActivatorVerifier method is provided in the RMI system of an activatable service to enable a client to verify a proxy's Activation ID. The call to getActivatorVerifier returns an activator verifier plus optional codebase and signer information. The activator verifier is an object received from the service containing code that implements the verifyActivatorTrust method, explained below. The codebase information is the location from which the code, i.e., the activator verifier, should have been

downloaded, e.g. a uniform resource locator (URL). The signer information identifies the creator or creators of the code, i.e., the activator verifier.

5 Because the verifier could have been downloaded, the verifyProxyTrust method checks that the activator verifier can be trusted using the codebase and signer information (step 416). If the service returned codebase information, the verifyProxyTrust method calls a local RMIClassLoader.getClassAnnotation method, which returns the location where the activator verifier code was obtained. The verifyProxyTrust method compares that location to the codebase information from the service. If they are different, the activator verifier is not used because its code is not trusted. If one or more signers is specified by the service, the verifyProxyTrust method obtains the signers of the verifier by calling the local Class.getSigners method and compares them to the signers specified by the service. If they are different, the activator verifier is not used because its code is not trusted.

10
15 In this way, the verifyProxyTrust method uses local methods to confirm the activator verifier code, which may have been downloaded and therefore could be corrupted. If the service did not specify either codebase or signer information, the verifyProxyTrust method confirms that the activator verifier was not downloaded, i.e., that the activator verifier is local, and therefore trusted. To do this, the verifyProxyTrust method compares the classloader of the verifier's class to the context classloader of the current thread or an ancestor of the

context classloader. If they are the same, then the activator verifier was not downloaded, and can be trusted.

Once the verifyProxyTrust method ensures that the activator verifier can be trusted, as described above, it extracts the Activation ID from the stub and calls the verifyActivatorTrust method of the verifier, passing the Activation ID as a parameter (step 418). The verifyActivatorTrust method will return normally if the service trusts the activation ID passed to it (step 420). If any of these calls do not return normally, i.e., throws an exception, then the proxy is not a secure RMI activatable stub (step 422), otherwise trust is established (step 424).

Trusting a proxy other than a secure RMI stub

If the proxy is not a secure RMI stub, the client can still establish trust in the proxy. If the proxy's class has a method with the signature

ProxyTrust getSecureProxy(),

then the verifyProxyTrust method makes the following local calls:

proxy2 = proxy.getSecureProxy();
verifyProxyTrust(proxy 2, constraints).

In this manner, the proxy's verifyProxyTrust method is called recursively to get down to a secure RMI stub. The original verifyProxyTrust, i.e., the method called by the client program, then makes a call to proxy2's getProxyVerifier method.

In response, the remote service returns a proxy verifier plus optional codebase and signer information. The verifyProxyTrust method checks the proxy verifier code using the codebase and/or signer information in the same way as

described above with respect to the activator verifier. Once the verifyProxyTrust method ensures that the proxy verifier can be trusted, it makes a local call to the verifyProxyTrust method of the verifier, passing the proxy as a parameter. If any of these calls do not return normally, i.e., throws an exception, then the proxy is not trusted.

If the proxy is not a secure RMI stub and does not have the getSecureProxy method, then an ordered list of TrustVerifier instances is obtained from the local configuration database, as described above. The verifyProxyTrust method calls the trustedProxy method of each TrustVerifier instance, with the proxy and client constraints as parameters. The trustedProxy method returns true if the given proxy is known to be trusted to correctly implement the RemoteSecurity interface, and false otherwise. If any method returns true, then trust is established. If none returns true, trust is not established.

Although systems and methods consistent with the present invention are described as operating in a Jini™ system using the Java™ programming language, one skilled in the art will appreciate that the present invention can be practiced in other systems and other programming environments. Additionally, although aspects of the present invention are described as being stored in memory, one skilled in the art will appreciate that these aspects can also be stored on or read from other types of computer-readable media, such as

secondary storage devices, like hard disks, floppy disks, or CD-ROM; a carrier wave from the Internet; or other forms of RAM or ROM. Sun, Sun Microsystems, the Sun Logo, Java™, and Jini™ trademarks are trademarks or registered trademarks of Sun Microsystems Inc. in the United States and other countries.

5 Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.

10

CLAIMS

What is claimed is:

1. A method in a distributed system, comprising the steps of:
downloading code from a server;
determining a set of constraints to implement secure communication with
the server; and
using secure code to verify that the downloaded code will enforce the set
of constraints when the downloaded code is used to communicate with the
server.
2. The method of claim 1, further comprising the step of:
using the downloaded code to invoke a method on the server, wherein the
downloaded code enforces the set of constraints on the server.
3. A method in a distributed system for ensuring trustworthiness of a first
proxy, comprising the steps of:
downloading the first proxy containing code for communication purposes;
using the first proxy to obtain a second proxy containing code for
communication purposes;
determining whether the second proxy is trustworthy by using a
trustworthiness verification routine;

determining whether a server is trustworthy by using the second proxy
when it has been determined that the second proxy is trustworthy;

requesting the server to determine whether the first proxy is trustworthy by
using the second proxy when it has been determined that the server is
trustworthy; and

using the first proxy to invoke a method on the server when it has been
determined that the first proxy is trustworthy, that the second proxy is
trustworthy, and that the server is trustworthy.

4. The method of claim 3, wherein the requesting step further comprises the
substeps of:

receiving a trust verifier routine from the server;
receiving codebase information and signer information for the trust
verifier from the server;

determining whether the trust verifier routine is trustworthy using
the codebase information and the signer information; and

when it has been determined that the trust verifier routine is
trustworthy, using the trust verifier routine to determine whether the first proxy is
trustworthy.

5. A method for establishing trust in a proxy containing code downloaded from a server, comprising the steps of:

determining whether the proxy is an instance of a trusted proxy class;

verifying at least one component of the proxy when it has been

determined that the proxy is an instance of the trusted proxy class, wherein the verifying step comprises the substeps of:

verifying trust in an invocation handler of the proxy;

determining whether the proxy has an activator; and

verifying the trustworthiness of the activator, when it has been

determined that the proxy has an activator; and

using the proxy to invoke a method on the server when it has been

determined that the proxy is an instance of the trusted class and the at least one component of the proxy has been verified successfully.

6. A method for establishing trust in a proxy containing code downloaded from a server, comprising the steps of:

determining whether the proxy is an instance of a trusted proxy class;

verifying at least one component of the proxy when it has been

determined that the proxy is an instance of a trusted proxy class, wherein the

proxy has an invocation handler and a plurality of socket factories, and wherein

the verifying step comprises the substeps of:

obtaining the invocation handler from the proxy;

10
15

20

LAW OFFICES

FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

testing whether the invocation handler is an instance of a secure invocation handler class;

comparing a class of each socket factory of the invocation handler to a list of trusted socket factory classes;

5 setting an error flag if the class of any socket factory of the invocation handler does not match the list of trusted socket factory classes;

determining whether the proxy has an activator; and

10 authenticating the activator, when it has been determined that the proxy has an activator, wherein the authenticating step further includes the substeps of:

obtaining an activator verifier from the server;

using the activator verifier to determine whether the activator is trusted by the server; and

15 setting the error flag, when it is determined that the activator is not trusted by the server; and

using the proxy to invoke a method on the server when the error flag is not set and when it has been determined that the proxy is an instance of the trusted class.

20 7. A distributed system comprising:

a server computer, comprising:

a memory with a service; and

a processor that runs the service; and

a client computer, comprising:

a memory with a proxy that facilitates use of the service, a client program that invokes a method of the service using the proxy, and a secure verifier that can be used to verify that the proxy will enforce security constraints when communicating with the service; and

a processor that runs the client program.

8. The distributed system of claim 7, wherein the server computer and the client computer communicate via the Internet.

9. The distributed system of claim 7, wherein the server computer and the client computer communicate via a local area network.

10. The distributed system of claim 7, wherein the security constraints are set by the client program.

11. The distributed system of claim 7, wherein the security constraints are set by the service.

12. A computer-readable medium containing instructions for controlling a data processing system to perform a method in a distributed system, the method comprising the steps of:

downloading code from a server;

determining a set of constraints to implement secure communication with the server; and

using secure code to verify that the downloaded code will enforce the set of constraints when the downloaded code is used to communicate with the server.

13. The computer-readable medium of claim 12, wherein the method further comprises the step of:

using the downloaded code to invoke a method on the server, wherein the downloaded code enforces the set of constraints on the server.

14. A computer-readable medium containing instructions for controlling a data processing system to perform a method in a distributed system the method comprising the steps of:

downloading the first proxy containing code for communication purposes;

using the first proxy to obtain a second proxy containing code for communication purposes;

determining whether the second proxy is trustworthy by using a
trustworthiness verification routine;

determining whether a server is trustworthy by using the second proxy
when it has been determined that the second proxy is trustworthy;

5 requesting the server to determine whether the first proxy is trustworthy by
using the second proxy when it has been determined that the server is
trustworthy; and

10 using the first proxy to invoke a method on the server when it has been
determined that the first proxy is trustworthy, that the second proxy is
trustworthy, and that the server is trustworthy.

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199

ABSTRACT

A system consistent with the present invention enables a program in a distributed system to determine whether downloaded code is trustworthy before using the downloaded code to communicate with other programs or services in the distributed system. A client that downloads proxy code from a service can verify that both the service and the downloaded code are trustworthy before using the code to communicate with the service. "Trustworthy" code is code the client knows will enforce the client's security constraints in communicating with the service, e.g., mutual authentication, confidentiality, and integrity.

5

10

11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

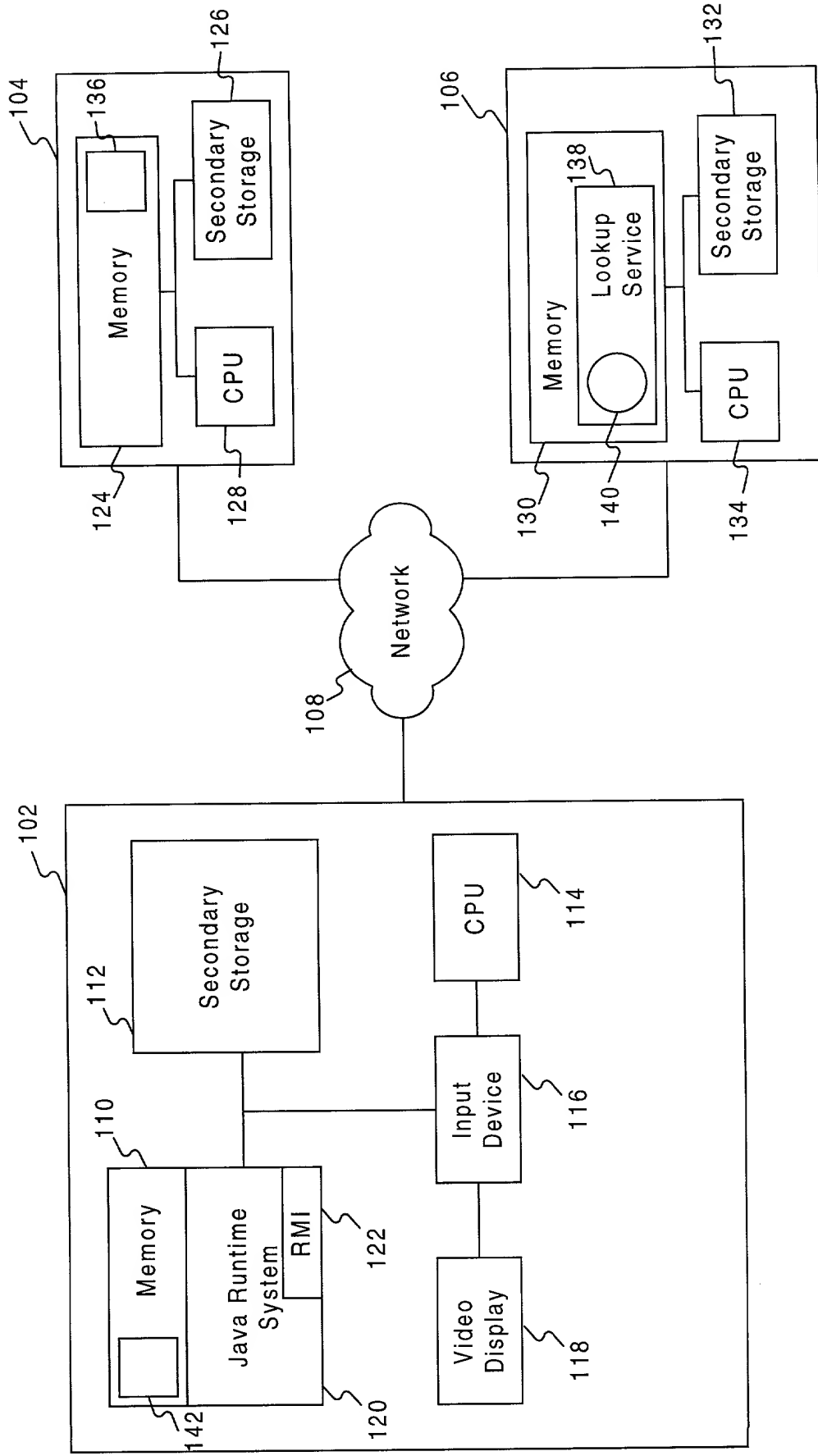


FIG. 1

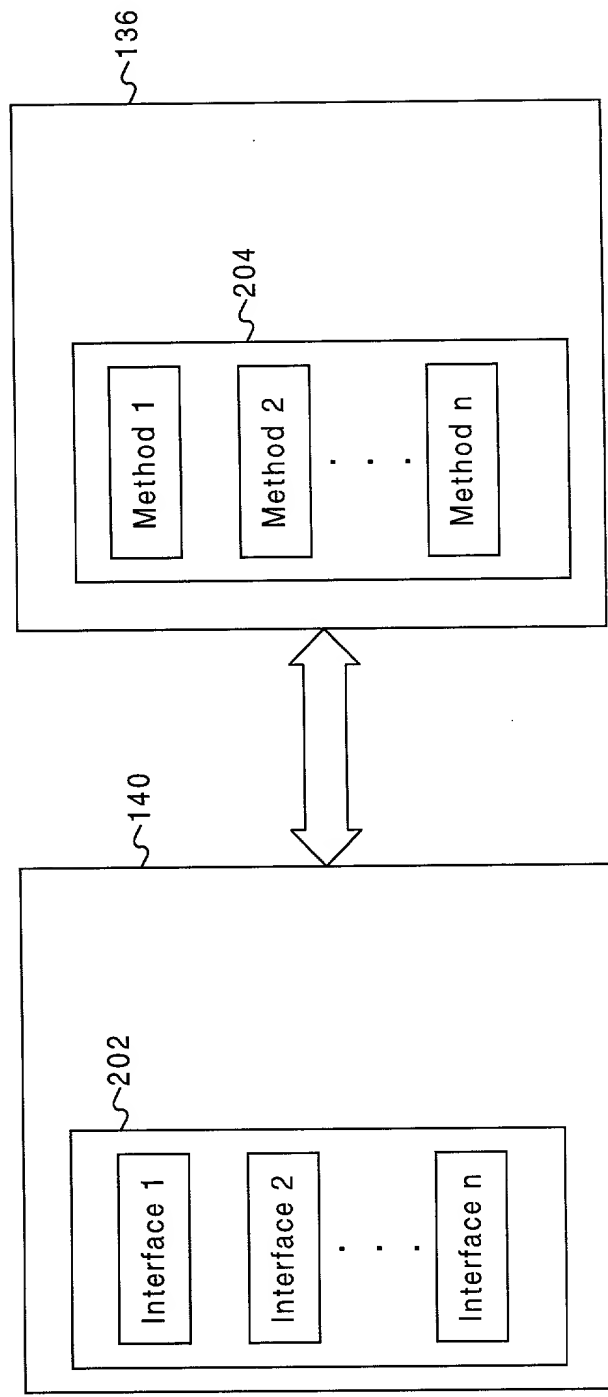


FIG. 2

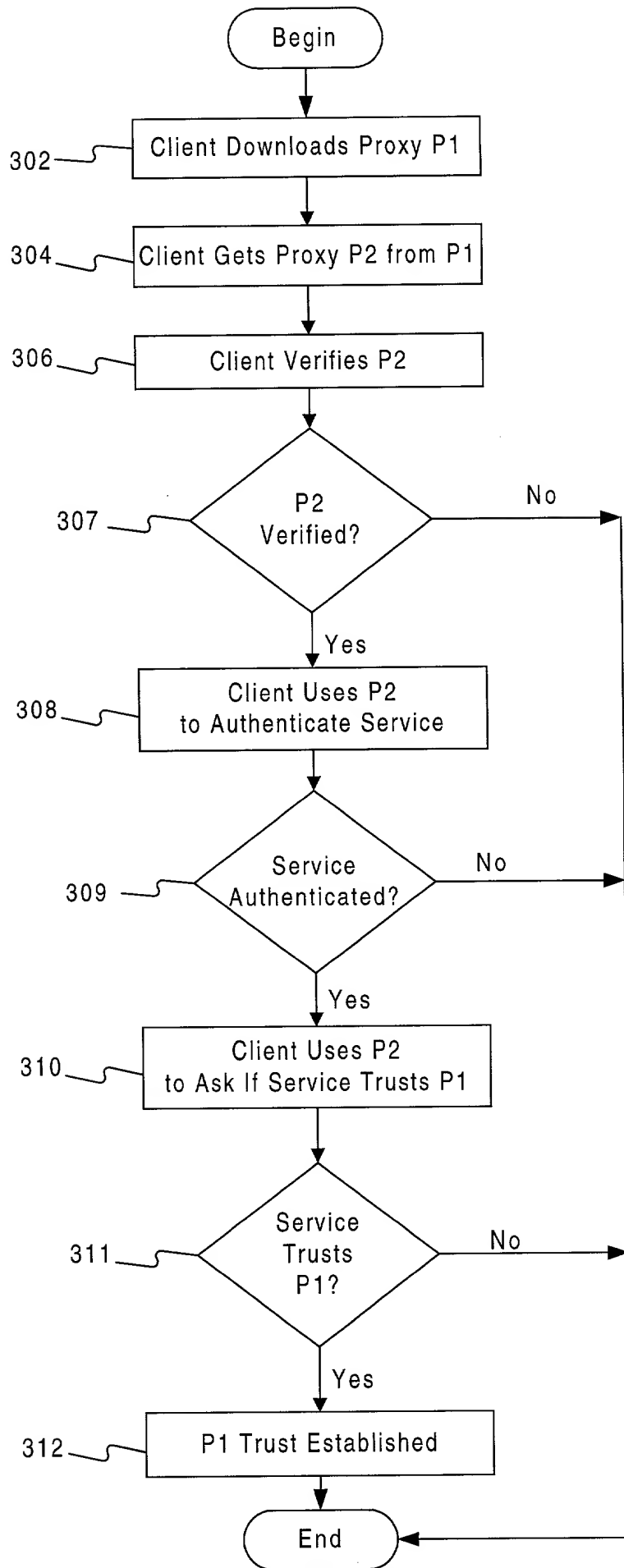


FIG. 3

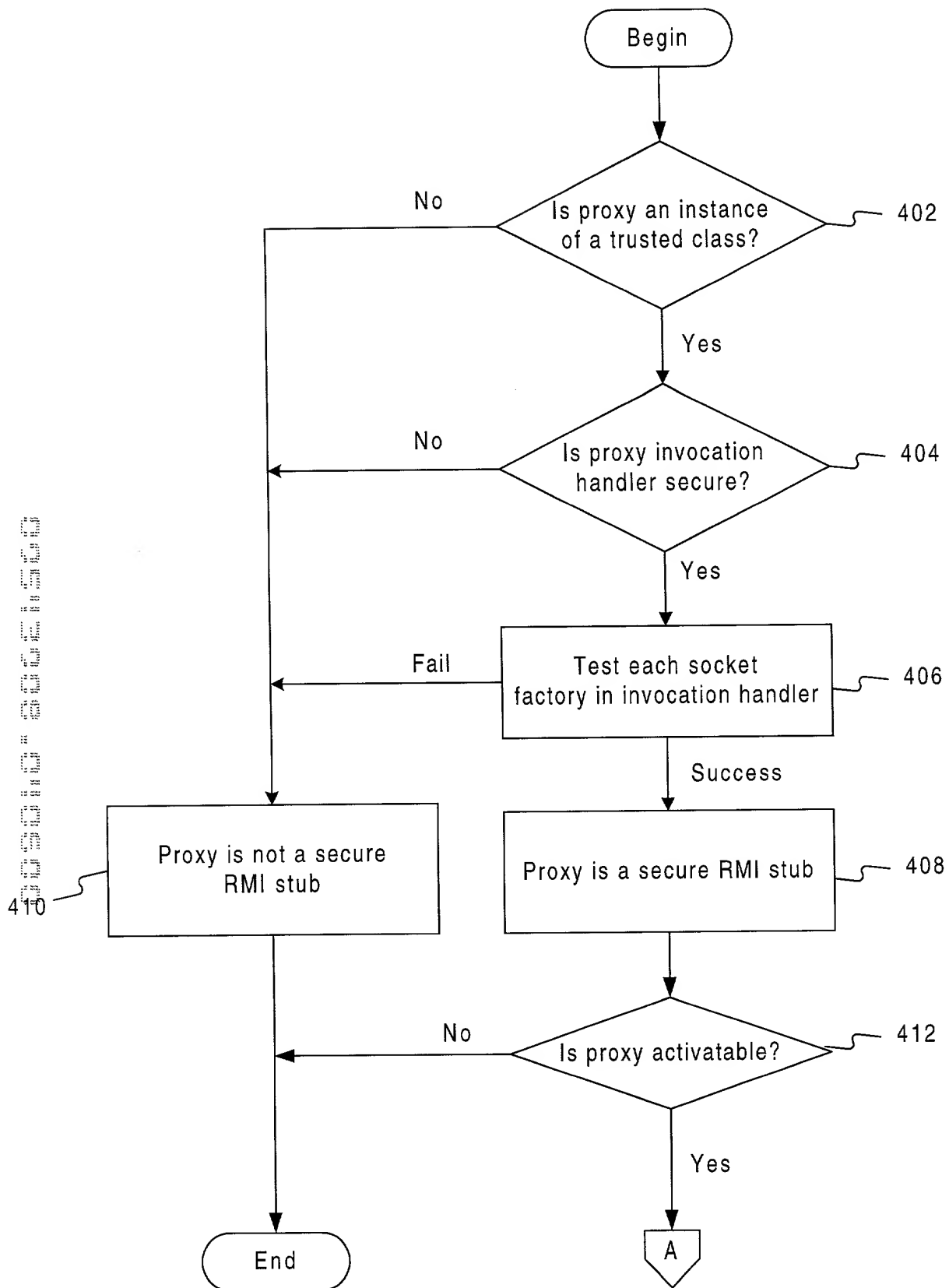


FIG. 4A

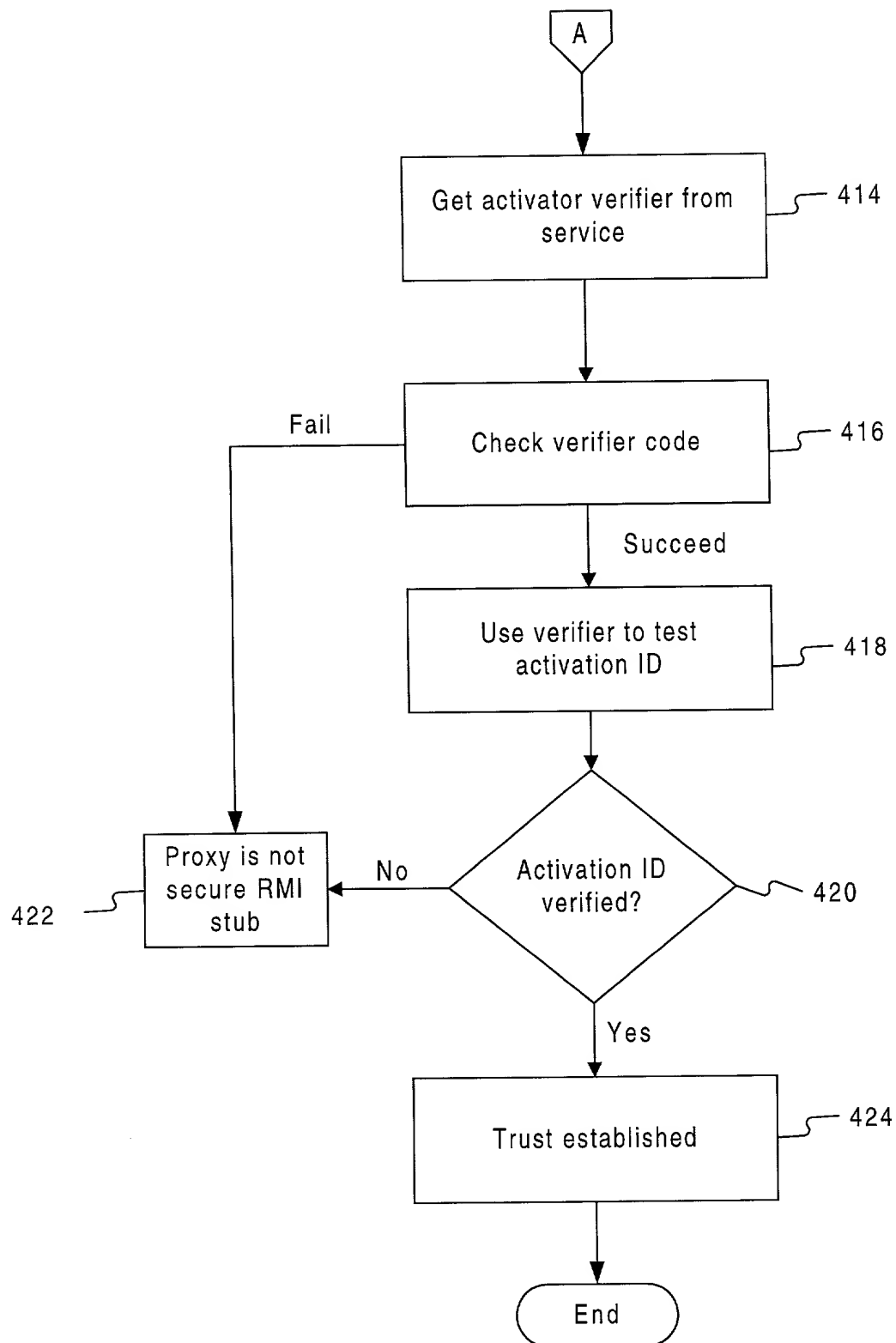


FIG. 4B